บทที่ 2

ระบบอินพุทเอาท์พุทในยูนิกซ์

ในบทนี้จะขอกล่าวถึงการจัดการอินพุทเอาท์พุทในระบบปฏิบัติการยูนิกซ์พร้อมทั้งโปรแกรมย่อยควบคุมอุปกรณ์อินพุทเอาท์พุท โดยเน้นถึงการทำงานของโปรแกรมย่อยควบคุมเทอร์มินัลเป็นหลัก

การจัดการกับอุปกรณ์อินพุทเอาท์พุทบนเครื่องคอมพิวเตอร์ต่างๆนั้น มักจะแตกต่างกันไปตามระบบปฏิบัติการ เช่นบางระบบมีโมดูลพิเศษที่อยู่นอกเคอร์เนล (kernel) ทำหน้าที่กำหนดและดูแลรักษาระบบแฟ้มข้อมูล แต่สำหรับยูนิกซ์การทำอินพุทเอาท์พุทบนแฟ้มหรือบนอุปกรณ์ต่างๆ นั้นกลับกลายเป็นหน้าที่ของเคอร์เนล การที่ออกแบบเช่นนี้มีวัตถุประสงค์เพื่อให้ลดความแตกต่างที่เกิดขึ้นระหว่างการทำอินพุทเอาท์พุทบนอุปกรณ์ที่จัดเก็บแฟ้มข้อมูลกับการทำอินพุทเอาท์พุทบนอุปกรณ์ประเภทเทอร์มินัลและเครื่องพิมพ์ ทำให้โปรแกรมของผู้ใช้สามารถเรียกใช้อุปกรณ์ต่างๆในรูปแบบเดียวกัน

ระบบแฟ้ม (file system)

ระบบแฟ้มข้อมูลของยูนิกซ์นั้นประกอบด้วยแฟ้ม 3 ประเภท คือ

- แฟ้มธรรมดา (regular file)
- ไดเรคทอรี
- แฟ้มพิเศษ

แฟ้มธรรมดา

สำหรับแฟ้มข้อมูลที่อยู่ในยูนิกซ์จะมีลักษณะที่แตกต่างจากแฟ้มข้อมูลที่อยู่ในระบบปฏิบัติการอื่น คือข้อมูลแต่ละตัวจะอยู่เรียงลำดับกันไปไม่มีลักษณะของเรคคอร์ดหรือตัวจบแฟ้มข้อมูล (file terminator) แต่ภายในยูนิกซ์ก็เอื้อให้สามารถเข้าหาข้อมูลทั้งในลักษณะแบบเรียงลำดับ (sequential) กับแบบสุ่ม (random) โดยอาศัยตัวชี้ของแฟ้มข้อมูลช่วย การแทรกข้อมูลหรือการลบข้อมูลที่อยู่กลางๆแฟ้มข้อมูลนั้นไม่อาจทำได้ซึ่งถ้าหากจำเป็นผู้ใช้ก็ต้องสร้างแฟ้มข้อมูลนั้นขึ้นมาใหม่

แฟ้มข้อมูลเหล่านี้จะมีหมายเลขประจำแฟ้มอยู่เรียกว่า i-number ซึ่งจะเป็นตัวชี้ไปยังรายการในแอเรย์ที่เรียกว่า i-node ที่เก็บรายละเอียดของแฟ้มข้อมูลนั้นเช่น ประเภทของแฟ้ม ขนาดของแฟ้ม วันที่ที่สร้าง วันที่ที่ปรับปรุงครั้งสุดท้าย

ไดเรคทอรี

ไดเรคทอรีก็คือแฟ้มข้อมูลที่เก็บชื่อของแฟ้มข้อมูลต่างๆเอาไว้ (ซึ่งอาจเป็นชื่อของไดเรคทอรีก็ได้) เพื่อใช้ในการแบ่งแยกแฟ้มข้อมูลเหล่านั้นออกเป็นกลุ่ม โดยที่สามารถจะแสดงความสัมพันธ์ของกลุ่มเหล่านั้นได้ การที่ไดเรคทอรีสามารถมีไดเรคทอรีซ้อนอยู่ได้ ทำให้โครงสร้างของแฟ้มข้อมูลต่างๆอยู่ในลักษณะลำดับชั้น เช่นเดียวกับทรี ซึ่งไดเรคทอรีที่อยู่บนสุดคือรูท (/) ภายในไดเรคทอรีจะเก็บเพียง i-number (2 ไบท์) กับชื่อของแฟ้มข้อมูล (14 ไบท์) การเรียกใช้แฟ้มข้อมูลจะต้องระบุ pathname ซึ่งประกอบด้วยชื่อของไดเรคทอรีที่อยู่ตั้งแต่บนสุดเรียงลำดับลงมา ซึ่งแต่ละไดเรคทอรีต้องคั่นด้วยเครื่องหมาย / เช่น /dir1/dir2/file เมื่อทำการเรียกใช้แฟ้มใด เคอร์เนลจะหาค่าของ i-number ของไดเรคทอรีแรกจากรูทไดเรคทอรี เพื่อเอาไปค้นหาในตาราง i-node จากตัวอย่าง เคอร์เนลทำการค้นหาชื่อ dir1 จากรูทไดเรคทอรี ซึ่งจะได้ i-number แล้วนำไปค้นในตาราง i-node เพื่อหาตำแหน่งของ dir1 ในดิสก์ จากนั้นก็ค้นหาชื่อของ dir2 จาก dir1 ไดเรคทอรีได้ค่า i-number เพื่อนำไปค้นในตาราง i-node อีก ให้ได้ตำแหน่งของ dir2 ในดิสก์ จากนั้นก็ค้นหาชื่อ file ใน dir2 ได้ค่า i-number เพื่อนำไปค้นในตาราง i-node อีกได้ตำแหน่งของ file ในดิสก์

แฟ้มข้อมูลพิเศษ

แฟ้มข้อมูลนี้จะแตกต่างจากแฟ้มข้อมูลธรรมดาตรงที่ไม่ได้ทำการเก็บข้อมูล แต่แฟ้มข้อมูลประเภทนี้เป็นเพียงชื่อที่แทนอุปกรณ์ต่างๆที่ต่อเข้ากับระบบ เช่น ดิสค์ เทป เทอร์มินัล เครื่องพิมพ์ ซึ่งจะอยู่ในไดเรคทอรี /dev เช่น /dev/tty00 ซึ่งหมายถึงพอร์ตเทอร์มินัลหมายเลข 0 แฟ้มข้อมูลพิเศษในยูนิกซ์แบ่งได้ออกเป็น 2 ประเภทคือ บล็อคดีไวซ์ (block device) และ คาแรคเตอร์ดีไวซ์ (character device)

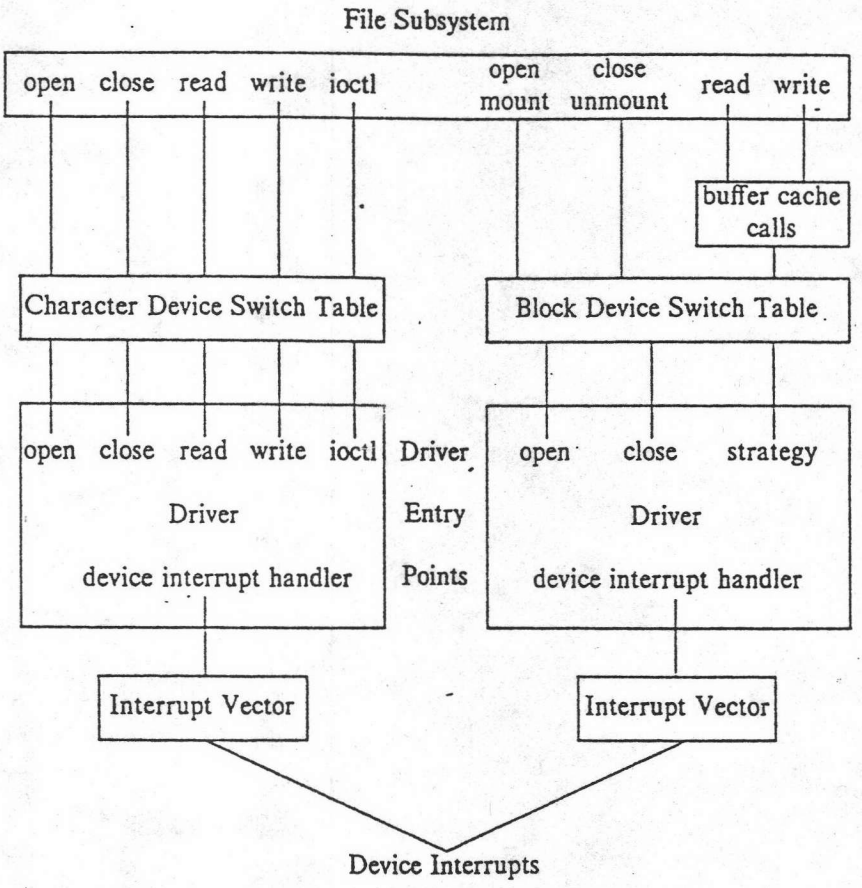
1. บล็อคดีไวซ์

อุปกรณ์ที่เป็นบล็อคดีไวซ์ได้แก่ ดิสค์ เทป การส่งผ่านข้อมูลจากอุปกรณ์ไปยังเคอร์เนลจะทำเป็นบล็อคซึ่งมีขนาดคงที่ (512 ไบท์ต่อ 1 บล็อค) โดยบนอุปกรณ์จะต้องมีบัฟเฟอร์ที่มีขนาดคงที่และในเคอร์เนลก็จำต้องมี pool ของบัฟเฟอร์ ใช้เป็น cache เพื่อเพิ่มความเร็วของอินพุทเอาท์พุท การเข้าถึงข้อมูลทำได้โดยแบบสุ่ม

2. คาแรคเตอร์ดีไวซ์

ได้แก่เทอร์มินัล เครื่องพิมพ์ ซึ่งอุปกรณ์เหล่านี้จะส่งผ่านข้อมูลแบบสายข้อมูล และไม่อาจเข้าถึงข้อมูลแบบสุ่มได้

สำหรับแฟ้มข้อมูลพิเศษก็จะมีรายการในตาราง i-node ซึ่งจะเก็บค่าของ device number แทนรายการจำนวนข้อมูลในแฟ้มของแฟ้มข้อมูลธรรมดา ซึ่งค่านี้จะถูกใช้โดยเคอร์เนลเป็นดัชนีชี้รายการในตาราง เพื่อหารูทีนของโปรแกรมย่อยควบคุมอุปกรณ์ ตารางที่กล่าวถึงนี้มีอยู่ 2 ตารางคือ block devices switch table กับ character devices switch table ซึ่งตารางทั้ง 2 นี้ทำหน้าที่เป็นอินเตอร์เฟซระหว่างเคอร์เนลกับโปรแกรมย่อยควบคุมอุปกรณ์ นั่นคือเมื่อมีการเรียก system call (เช่น read หรือ write) จากโปรแกรมของผู้ใช้ เคอร์เนลจะหาค่า device number ของอุปกรณ์ที่ต้องการอ่านหรือบันทึกจากตาราง i-node แล้วตรวจสอบว่าประเภทของแฟ้มข้อมูลเป็นประเภทใดเพื่อเลือกว่าจะหาตำแหน่งของรูทีนที่ทำหน้าที่ในการอ่านหรือบันทึกจาก block devices switch table หรือ character devices switch table



รูปที่ 2.1 เอนทรีของโปรแกรมย่อยควบคุมอุปกรณ์

โปรแกรมย่อยควบคุมอุปกรณ์

อาจกล่าวได้ว่าโปรแกรมย่อยควบคุมอุปกรณ์เป็นกลุ่มของโปรแกรมย่อยและข้อมูลที่อยู่ภายในเคอร์เนลโดยทำหน้าที่เป็นตัวเชื่อมโยงระหว่างโปรแกรมของผู้ใช้กับตัวอุปกรณ์ โปรแกรมย่อยนี้เขียนด้วยภาษาซี แล้วแปลเป็นโมดูลประสงค์ (object module) เพื่อจะเชื่อมโยงเข้ากับส่วนอื่นของเคอร์เนลด้วยโปรแกรมบรรจุ (loader) นั่นคือถ้าหากต้องการเขียนโปรแกรมย่อยควบคุมอุปกรณ์ขึ้นมาใหม่ หรือต้องการเปลี่ยนแปลงในตัวโปรแกรมย่อยควบคุมอุปกรณ์เหล่านั้น จะต้องเชื่อมโยงส่วนต่างๆใหม่เพื่อสร้างเคอร์เนลใหม่

การเรียกใช้โปรแกรมย่อยควบคุมอุปกรณ์ทำได้โดยเรียกผ่านเอนทรีพอยน์มาตรฐาน ซึ่งเอนทรีพอยน์จะแตกต่างกันไปตามประเภทและลักษณะของอุปกรณ์ (เช่น เครื่องพิมพ์ ก็ไม่จำเป็นต้องมีฟังก์ชั่นสำหรับการอ่านข้อมูล)

เอนทรีพอยน์ที่พบโดยทั่วไปในโปรแกรมย่อยควบคุมอุปกรณ์

- รูทีนการเริ่มต้นจะถูกเรียกเมื่ออยู่ในระหว่างการเริ่มเปิดระบบ
- รูทีนการเปิดและปิด
- รูทีนการอ่านและเขียน
- รูทีนที่จัดการกับการขัดจังหวะ
- รูทีนที่จัดการ synchronous i/o multiplexing

โปรแกรมย่อยควบคุมอุปกรณ์สามารถแบ่งได้ออกตามประเภทของอุปกรณ์ คือ โปรแกรมย่อยควบคุมอุปกรณ์แบบบล็อคและโปรแกรมย่อยควบคุมแบบคาแรคเตอร์ ซึ่งต่อไปจะขอกล่าวถึงลักษณะของโปรแกรมย่อยควบคุมเทอร์มินัลซึ่งเป็นโปรแกรมย่อยควบคุมแบบคาแรคเตอร์แบบหนึ่ง

โปรแกรมย่อยควบคุมเทอร์มินัล

สำหรับเทอร์มินัลนั้นถือว่าเป็นแฟ้มข้อมูลแบบพิเศษ ซึ่งโปรแกรมของผู้ใช้สามารถอ่านหรือบันทึกได้เช่นเดียวกับแฟ้มข้อมูลอื่นโดยผ่านชื่อของแฟ้มข้อมูล /dev/tty01 หรือ ผ่าน standard input หรือ standard output หรือ standard error ที่กำหนดให้อยู่แล้วสำหรับแต่ละโปรเซส หรือ ผ่าน /dev/tty

การส่งข้อมูลจากโปรแกรมไปแสดงบนจอภาพ จะถูกกรองโดยเคอร์เนล ก่อนที่จะส่งไปที่เทอร์มินัล ในทำนองเดียวกันการอ่านข้อมูลจากเทอร์มินัล ข้อมูลเหล่านั้นจะต้องผ่านเคอร์เนลก่อนที่จะส่งให้กับโปรแกรม

ส่วนของเคอร์เนลที่กล่าวถึงนี้ก็คือ โปรแกรมย่อยควบคุมเทอร์มินัล ซึ่งจะกล่าวถึงหน้าที่ของโปรแกรมย่อยควบคุมเทอร์มินัลอย่างคร่าวๆได้ดังนี้

- กรณีส่งตัวอักษรไปยังเทอร์มินัล (write)

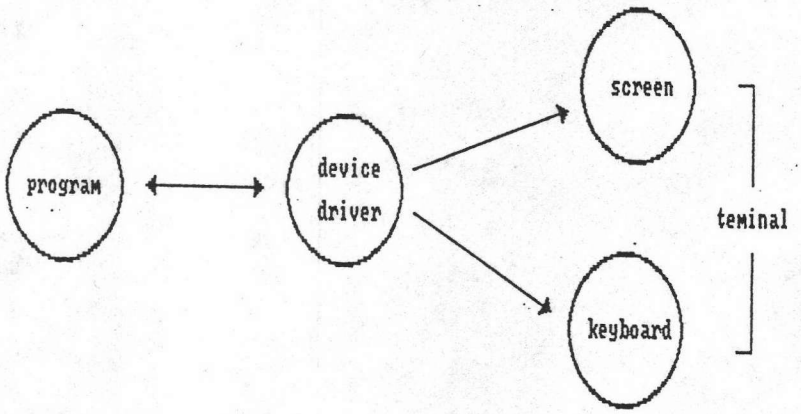
แปลงตัวอักษรเล็กเป็นตัวอักษรใหญ่ ขยายตัวอักษรกั้นวรรคเป็นช่องว่าง แปลงตัวอักษรขึ้นบรรทัดใหม่หรือรีเทอร์นเป็น ตัวอักษรขึ้นบรรทัดใหม่และรีเทอร์น กรองบิตที่ 8 ของแต่ละตัวอักษรเพื่อใช้เป็นบิตเสริม (parity bit) ในลักษณะ ภาวะเสริมคู่ (odd parity) หรือ ภาวะเสริมคี่ (even parity) หรือไม่มีภาวะเสริม

- กรณีรับตัวอักษรจากเทอร์มินัล (read)

แปลงตัวอักษรขึ้นบรรทัดใหม่และรีเทอร์นเป็นตัวอักษรขึ้นบรรทัดใหม่ ตรวจสอบภาวะเสริมของตัวอักษรแต่ละตัวที่เข้ามา ไม่รับตัวอักษรที่มีภาวะเสริมผิด นอกจากนี้แปลงตัวอักษรบางตัวเป็นการกระทำแบบพิเศษ และไม่ส่งให้กับโปรแกรม เช่น ctrl-S ใช้หยุดการแสดงผลบนเทอร์มินัล ctrl-Q ให้เริ่มแสดงผลต่อ ctrl-C เป็นอินเทอร์รัพท์ เพื่อเลิกการทำงาน ctrl-D ใช้เพื่อบอกว่าจบข้อมูลที่มาจากแป้นอักษรของเทอร์มินัล

นอกจากนี้โปรแกรมย่อยควบคุมเทอร์มินัลยังจัดการทางด้านฮาร์ดแวร์ เช่น อัตราความเร็วในการส่งข้อมูล ภาวะเสริม และสามารถควบคุมการไหลของข้อมูล (flow control)

สิ่งต่างๆที่ได้กล่าวนี้สามารถเปลี่ยนแปลงได้โดยผ่านคำสั่ง stty จากเชลล์หรือ system call ชื่อ ioctl จากโปรแกรมของผู้ใช้ ซึ่งจะกล่าวถึงต่อไปในภายหลัง

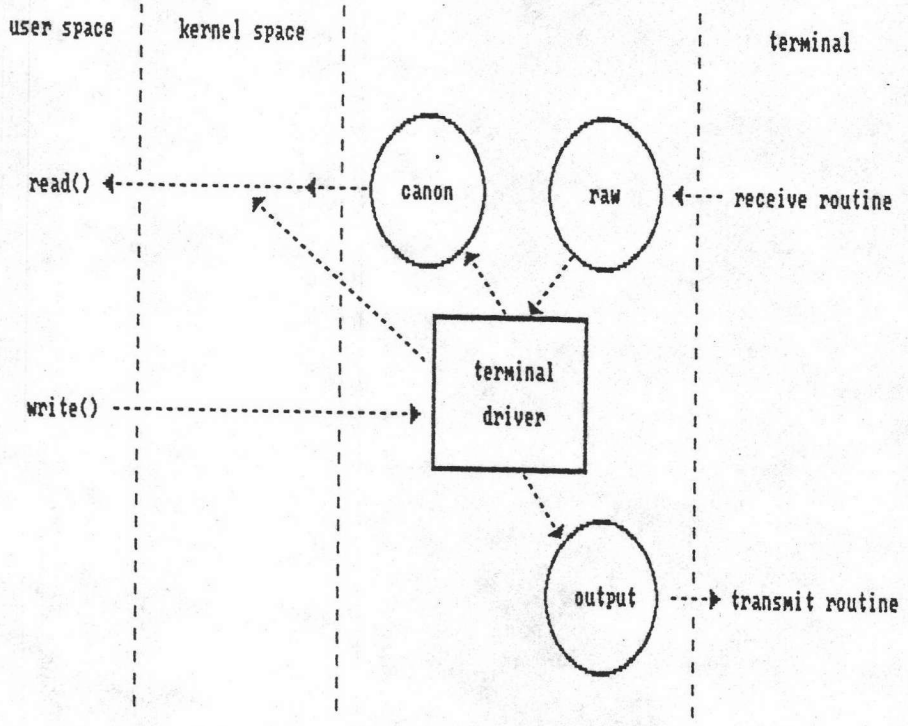


รูปที่ 2.2 ความสัมพันธ์ระหว่างระบบยูนิกซ์กับเทอร์มินัล

จากรูปที่ 2.2 เมื่อโปรแกรมของผู้ใช้เรียกฟังก์ชั่นเขียน (write) โปรแกรมย่อยควบคุมเทอร์มินัลจะนำเอาข้อมูลเหล่านั้นไปแสดงบนจอภาพพร้อมทั้งตีความตัวอักขระพิเศษ (เช่น กั้นวรรค ขึ้นบรรทัดใหม่) ในทำนองเดียวกันเมื่อโปรแกรมเรียกฟังก์ชั่นอ่านโปรแกรมนี้จะอ่านข้อมูลจากแป้นพิมพ์ พร้อมทั้งแสดงข้อมูล (echo) เหล่านั้นออกทางจอภาพ แต่ข้อมูลจะยังไม่ส่งให้กับโปรแกรมของผู้ใช้จนกว่าผู้ใช้กดแป้นตัวอักษรรีเทอร์น (return)

การทำงานระหว่างโปรแกรมย่อยควบคุมอุปกรณ์กับเทอร์มินัลเป็นแบบฟูลดูเพล็ก ดังนั้นการแสดงผลออกทางจอภาพจะเป็นหน้าที่ของโปรแกรมย่อยควบคุมเทอร์มินัล มิได้เป็นหน้าที่ของเทอร์มินัล เมื่อมีการป้อนข้อมูลเข้ามาทีละอักขระ ข้อมูลก็จะถูกส่งไปให้โปรแกรมย่อยควบคุมเทอร์มินัล ซึ่งจะทำสำเนาข้อมูลเหล่านั้นแล้วแสดงออกทางจอภาพ แต่ข้อมูลจะยังไม่ถูกส่งไปให้กับโปรแกรมของผู้ใช้ จนกว่าโปรแกรมของผู้ใช้พร้อมที่จะอ่านข้อมูล ซึ่งถ้าหากว่าในขณะที่โปรแกรมผู้ใช้สั่งเขียนออกบนจอภาพ เป็นเวลาเดียวกันกับที่ผู้ใช้ป้อนข้อมูลเข้ามาก็จะแสดงออกทางจอภาพปนกับข้อมูลที่สั่งเขียนจากโปรแกรม และเมื่อโปรแกรมสั่งอ่านข้อมูลข้อมูลที่ถูกป้อนเข้ามาเหล่านั้นจะถูกส่งไปให้กับโปรแกรม การที่โปรแกรมย่อยควบคุมเทอร์มินัลสามารถทำงานในลักษณะนี้ได้ ต้องอาศัยบัฟเฟอร์ที่มีโครงสร้างเป็นรายการเชื่อมโยง (linked list) ซึ่งมีอยู่ด้วยกัน 3 ชุด คือ รอว์บัฟเฟอร์ คานอนบัฟเฟอร์ และเอาท์พุทบัฟเฟอร์แต่การทำงานของโปรแกรมย่อยควบคุมเทอร์มินัลยังจะขึ้นอยู่กับสภาวะของเทอร์มินัล ซึ่งมีอยู่ด้วยกัน 2 สภาวะ คือ

คาโนนิคัลโหมด และรอว์โหมด โดยจะมีผลต่อการแสดงผลบนจอภาพและข้อมูลที่จะส่งให้กับโปรแกรมของผู้ใช้ เช่น ต้องการอ่านข้อมูลจากแป้นพิมพ์ขณะที่เทอร์มินัลอยู่ในสภาวะคาโนนิคัลโหมด ถ้าหากผู้ใช้ต้องการพิมพ์ตัวอักษร d a t e แต่ป้อนเป็น d s t e แทน ผู้ใช้ต้องป้อนตัว backspace 3 ตัว และ ตัว a t e ใหม่ ซึ่งในที่สุดโปรแกรมของผู้ใช้จะได้ตัวอักษร 4 ตัว คือ date แต่ถ้าหากเทอร์มินัลอยู่ในสภาวะรอว์โหมด โปรแกรมของผู้ใช้จะได้ ตัวอักษรทั้งหมด 11 ตัว ลักษณะการส่งผ่านข้อมูลจากเทอร์มินัลไปยังโปรแกรมของผู้ใช้ แสดงได้ดังรูปที่ 2.3



รูปที่ 2.3 การไหลของข้อมูลในโปรแกรมย่อยควบคุมเทอร์มินัล

<u>เทอร์มินัลในคาโนนิคัลโหมด</u>

เมื่อโปรแกรมของผู้ใช้เรียกฟังก์ชั่น write() ข้อมูลที่อยู่ใน user space จะถูกส่งไปให้โปรแกรมย่อยควบคุมเทอร์มินัล เพื่อที่จะส่งและแปลงตัวอักขระพิเศษ (เช่น กั้นวรรคจะถูกแปลงเป็นช่องว่างหลายตัวติดกัน) ไปเก็บไว้ในเอาท์พุทบัฟเฟอร์ จนกว่าข้อมูลที่อยู่ในเอาท์พุทบัฟเฟอร์มีมากจนถึงจุดที่เรียกว่า high-water mark จึงเรียกโปรแกรมย่อยควบคุมที่ทำหน้าที่เอาผลออกแสดงทางจอภาพให้ทำงาน ซึ่งในขณะนี้โปรแกรมย่อยควบคุมเทอร์มินัลก็จะถูกปลุกให้ขึ้นมาทำงานต่อเป็นอย่างนี้เรื่อยไปจนกว่าข้อมูลที่ส่งมาจากโปรแกรมของผู้ใช้หมด

ส่วนการทำงานของโปรแกรมย่อยควบคุมเทอร์มินัลเมื่อเรียกฟังก์ชั่น read() ค่อนข้างจะซับซ้อนมากกว่าขั้นตอนการ write() โดยหลังจากที่ถูกเรียกฟังก์ชั่น read() จะตรวจสอบในรอว์บัฟเฟอร์ว่ามีข้อมูลหรือไม่ ถ้าหากไม่มีโปรแกรมย่อยนี้ก็จะหลับจนกว่าจะได้รับการแจ้งจาก terminal interrupt handler ส่วน teminal interrupt handler ถูกปลุกด้วยสัญญาณการขัดจังหวะจาก teminal controller ซึ่งจะเก็บข้อมูลในรอว์บัฟเฟอร์ไปเรื่อยๆพร้อมทั้งส่งข้อมูลไปยังเอาท์พุทบัฟเฟอร์เพื่อแสดงตัวอักษรที่ถูกป้อนเข้ามาออกทางจอภาพ โดยจะแปลงตัวอักขระพิเศษต่างๆ (เช่น erase ถูกแปลงเป็นตัวอักษร 3ตัว คือ backspace space backspace) ให้อยู่ในรูปเบบที่ผู้ใช้เข้าใจได้ แต่ถ้าหากอักขระพิเศษที่ป้อนเข้ามาคือตัวปัดแคร่ (carriage return) โปรแกรมย่อยนี้จะหยุดการดึงข้อมูลจากเทอร์มินัลใส่ลงในรอว์บัฟเฟอร์ แต่จะดึงข้อมูลจากรอว์บัฟเฟอร์ไปใส่ในคานอนบัฟเฟอร์แทน โดยจะแปลงตัวอักขระพิเศษ (เช่น erase ก็จะลบข้อมูลที่อยู่ในคานอนบัฟเฟอร์ 1 ตัว หรือ ตัวปัดแคร่จะถูกแปลงเป็นตัวอักษรขึ้นบรรทัดใหม่) เพื่อส่งให้กับโปรแกรมของผู้ใช้หลังจากที่ดึงข้อมูลจากรอว์บัฟเฟอร์มาตรวจสอบจนหมด

เทอร์มินัลในรอว์โหมด

สำหรับในรอว์โหมดนั้นข้อมูลจะถูกส่งไปให้โปรแกรมของผู้ใช้เพียง 1 ตัวอักษรเมื่อทำการเรียกฟังก์ชั่น read() 1 ครั้ง (กรณีนี้เป็นจริงเฉพาะระบบปฏิบัติการยูนิกซ์ตระกูลบีเอสดีเท่านั้น) ตัวอักษรทุกตัวจะถูกเก็บลงในรอว์บัฟเฟอร์ แล้วส่งให้กับโปรแกรมของผู้ใช้พร้อมทั้งแสดงออกทางจอภาพโดยไม่มีการแปลงตัวอักขระพิเศษดังเช่นในคาโนนิคัลโหมด ทำให้ตัวปัดแคร่ถูกมองเป็นข้อมูลให้กับโปรแกรมของผู้ใช้แทนการมองเป็นตัวจบบรรทัดของการป้อนข้อมูล ลักษณะเช่นนี้จะทำให้เหมาะกับการเขียนโปรแกรมที่ต้องการอ่านข้อมูลโดยไม่ต้องรอเคาะตัวปัดแคร่ เช่น โปรแกรมบรรณาธิการ vi สามารถใช้คำสั่งเป็นตัวอักษรใดๆแทนพวกคอนโทรลคาแรคเตอร์